INTRODUCTION

Un programme informatique se compose d'une série d'instructions, qui sont exécutées successivement par l'ordinateur. Lors de la création d'un programme, il est nécessaire d'y inclure les instructions correspondant à la tâche qu'on souhaite voir accomplie par l'ordinateur. Le processus consistant à définir les instructions, devant être exécutées par l'ordinateur, se nomme ***programmation***.

Beaucoup de langages de programmation sont apparus au fil des années en commençant par les langages intimement liés à la machine (langage machine et assembleur), viennent après les langages dits évolués : interprétés tels que Basic et compilés tels que ***Pascal*** et ***C***.

Langage Pascal

Le langage Pascal est un langage de programmation de haut niveau. Il a été conçu par le professeur **WIRTH** à Zurich en 1970 comme une aide à l'enseignement de ***la programmation structurée*.**

Langage C

Le langage C a été créé en 1972 par **Denis RITCHIE** avec un objectif relativement limité : écrire un système d'exploitation (UNIX). Mais, ses qualités opérationnelles l'ont très vite fait adopté par une large communauté de programmeurs.

Une première version du langage est apparue en 1978 avec l'ouvrage de Kernighan et Ritchie « ***The C programming language*** ». Mais le langage a continué d'évoluer après cette date à travers les différents compilateurs qui ont vu le jour. Son succès internationnel a amené l'ANSI (American National Standard Institute) à définir un C standard (on le qualifie souvent par le C ANSI).

Choix du sujet

L'enseignement de l'informatique se répand de plus en plus dans les établissements scolaires et pour toutes les options, à l'université, aux écoles d'ingénieurs, plus récemment dans les lycées et collèges et même dans les écoles primaires privées.

En particulier, l'enseignement de l'algorithmique et de la programmation est, et reste toujours un passage obligatoire pour la formation des futurs informaticiens. Malgré l'apparition d'autres systèmes de programmation très évolués et reposant sur la programmation orientée objets, ceux-ci restent liés aux langages de programmation de base tels que Pascal et C.

Objectif

Notre objectif c'est de palier aux difficultés que le programmeur (universitaire, élève ingénieur, analyste-programmeur, etc.) puisse rencontrer au moment de son apprentissage en lui proposant une série d'exercices diversifiés et couvrant la totalité des notions de base de la programmation. On lui propose aussi, à la fin du document, toutes les solutions des exercices proposés en Pascal et en C.

ÉLÉMENTS DE BASE D'UN PROGRAMME EN PASCAL

I- Structure d'un programme

Un programme PASCAl se compose des éléments suivants :

1- En-tête qui se compose du mot réservé PROGRAM suivi d'un identificateur du nom du programme, suivi du point virgule ' ;'.

Exemple : PROGRAM Operations ;

2- Préambule déclaratif qui comporte :

- La déclaration des bibliothèques utilisées, précédée par le mot réservé USES. Les identificateurs des bibliothèques sont séparés par des virgules et la liste est terminée par un point virgule ';'.

  Exemple : USES Crt, Graph, Dos ;

- La déclaration des identificateurs de types de données, définis par l'utilisateur, précédée par le mot réservé TYPE.

  Exemple : TYPE Nombre = 0..99 ;

- La déclaration des identificateurs de constantes, précédée par le mot réservé CONST.

  Exemple : CONST Pi = 3.14 ;

- La déclaration des identificateurs de variables, précédée par le mot réservé VAR.

  Exemple : VAR A : INTEGER ;

3- Définition des différents sous-programmes (voir chapitre des sous-programmes).

4- Définition du bloc principal du programme : ce bloc commence par le mot réservé BEGIN suivi d'une séquence d'instructions (séparées par des points

virgules ' ;') et se termine par le mot réservé END suivi du point '**.**'.

II- Types de données

II-1. Variables et constantes

II-1.1 Les variables

On appelle variable, un emplacement mémoire dans lequel est codée une information que l'on peut modifier et utiliser grâce à un identificateur. Toute variable doit être déclarée avant d'être utilisée.

Syntaxe : Var <identificateur> : <type>;

Exemple : var x : integer ;

Variables globales et variables locales

Les variables déclarées dans la partie déclaration du programme sont dites **globales**.
Les variables déclarées dans la partie déclaration d'un sous-programme sont dites **locales**.
Les variables globales sont connues par tous les sous-programmes, tandis que les variables locales sont connues seulement par le sous-programme dans lequel elles sont déclarées.

II-1.2 Les constantes

Contrairement à une variable, une constante ne peut être modifiée (théoriquement).
Syntaxe : const <identificateur> = valeur ;

Exemple :

Remarque

Contrairement aux variables, il n'est nullement besoin de spécifier le type de la constante. On peut tout de même le faire, en utilisant le double point, comme pour les variables.

Exemple : const Gravite = 9.81;

II-2. Les types de base

Type	Description	Intervalle	Exemples	Mémoire requise
Shortint	Entiers courts	-128 à 127	-125; 0; 32	1 octet
Integer	Entiers "relatifs"	-32 768 à 32 767	-30 000; 421;	2 octets
Longint	Entiers longs	-2147483648 à 2147483647	-12 545 454; 3 257	4 octets
Byte	Entiers sur 1 Bit (Byte ou Octet)	0 à 255	12; 157	1 octet
Word	Entiers sur 2 Bits (Word ou Mot)	0 à 65 535	27; 4 589	2 octets
Real	Nombres réels	$2.9E^{-39}$ à $1.7E^{38}$	3.1415; 789.457851	6 octets
Single	Nombres décimaux (simple précision)	$1.5E^{-45}$ à $3.4E^{38}$	3.1415926; 178 925.455678	4 octets
Double	Nombres décimaux (double précision)	$5E^{-324}$ à $1.7E^{308}$	54.5899; 9 897 669	8 octets

Type	Description	Intervalle	Exemples	Mémoire requise
			651.45568959	
Extended	Nombres réels	$3.4E^{-4932}$ à $1.1E^{4932}$	3.14159265458; 9.81	10 octets
Comp	Entier	$-9.2E^{18}$ à $9.2E^{18}$	-271; 6 548	8 octets
Boolean	logique sur 1 octet	false ou true	false; true	1 octet
String	Chaîne de caractères	256 caractères au maximum (0 à 255)	'Hello!'; 'Allez-vous bien ?'	256 octets
String[n]	Chaîne de n caractères	n caractères maximum	String[6]->'Hello!'	n octets
Char	1 caractère	1 caractère maximum	'R'	1 octet

II-3. Créer un type de données

Il est possible au programmeur de créer ses propres types. Les types doivent être déclarés avec le mot-clef ***Type*** à fortiori avant la déclaration des variables.

Exemple

Type matrice = Array[1..10,1..10] of integer;
Définit un type nommé matrice pour représenter une matrice carrée d'ordre 10.

Type énuméré

Dans la déclaration d'un type énuméré, il faut énumérer toutes les valeurs de ce type.

Syntaxe : type nom_du_type = (identificateur_1, identificateur_2,.., identificateur_n)

Exemple : type jour = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche) ;

Type intervalle

Syntaxe : type nom_du_type = début_de_l_intervalle..fin_de_l_intervalle

Exemple : Type age = 1..150 ;

Déclare un type nommé age dont les valeurs sont des entiers compris entre 1 et 150.

III- Opérateurs et expressions

Les expressions sont composées d'opérateurs et d'opérandes. La plupart des opérateurs sont binaires, c'est à dire qu'ils mettent en œuvre deux opérandes (exemple A+B). Les opérateurs à un opérande sont dits unaires (exemple –A). Dans les expressions plus complexes, l'existence de règles de priorité permet d'éliminer toute ambiguïté dans l'ordre de réalisation des opérations.

Table de priorité des opérateurs

Ordre	Opérateurs	Ordre	Opérateurs
0	(, [	5	+ , -
1	OR	6	*, /, DIV, MOD
2	AND	7	(+) , (-)
3	NOT	8	
4	< , >, <= , >= , <> , =	9	

Remarque : (+) et (-) deux opérateurs unaires.

Les trois règles fondamentales de priorité des opérateurs sont les suivantes :

1- Un opérande placé entre deux opérateurs de priorités différentes sera lié à celui possédant la priorité la plus élevée.

2- Un opérande placé entre deux opérateurs de même priorité sera lié à celui qui se trouve à gauche.

3- Les expressions contenues entre parenthèses sont évaluées d'abord afin de traiter leur résultat comme un seul opérande.

Dans une expression arithmétique, les opérateurs sont les opérateurs arithmétiques (+,-,*, /, DIV, MOD).

Un opérande peut être :

- Un nom de variable ou de constante numérique.
- Une constante numérique.
- Un nom de fonction de type numérique tel que COS, SIN, etc.

Une expression logique simple est une comparaison entre deux expressions arithmétiques. Les opérateurs de comparaison sont =, <>, <, >, <=, >=.

Une expression logique est la composée d'expressions logiques simples par les opérateurs logiques :

- OR et AND opérateurs logiques binaires, OR pour la disjonction et AND pour la conjonction.

- NOT opérateur unaire, NOT opérateur de négation.

IV- Instructions simples

IV-1. Instructions d'entrée

Une instruction d'entrée permet de lire une donnée à partir du clavier.

Syntaxe :

Read (V_1, V2,…., Vn) Où V_i est une variable de tout type simple (sauf énuméré) ou de type chaîne de caractères.

ReadLn (V_1,V_2,...V_n) même effet que Read, mais il faut taper la touche Entrée après l'entrée de la donnée. L'instruction ReadLn (sans argument) attend la frappe de la touche Entrée au clavier.

Exemple : read(x) ;

IV-2. Instructions de sortie

Une instruction de sortie permet l'affichage sur l'écran des valeurs correspondantes aux arguments considérés.

Syntaxe

Write (val1,val2,…,valn) Où vali est une valeur d'une donnée constante ou variable (sauf type énuméré), une constante chaîne de caractères, ou une valeur d'une expression.

WriteLn(val_1,val_2,…val_n) même effet que Write, mais le curseur passe à la ligne suivante après l'affichage.

Exemple : write('salut ',nom) ;

Remarque :

- Une constante chaîne de caractères est mise entre apostrophe. Une apostrophe comprise dans une chaîne doit être dédoublée.
- L'instruction WriteLn (sans arguments) permet un saut de ligne.

IV-3. Instruction d'affectation

Cette instruction permet de transcrire une valeur dans une variable. Le symbole d'affectation est ' := '

Syntaxe : <Var> := <Val> ;

Où la variable <Var> peut être de tout type et la valeur <Val> peut être une valeur constante, la valeur d'une donnée constante ou variable, la valeur d'une fonction, ou le résultat d'une expression.

Exemple : x :=2 ;

V-Enoncé des exercices

Exercice 1
Ecrire un programme qui permet d'afficher le message suivant : mon premier programme.

Exercice 2
Ecrire un programme qui demande à l'utilisateur les valeurs de 2 entiers x et y, qui permute leurs valeurs et qui les affiche.

Exercice 3
Ecrire un programme qui échange 3 entiers.

Exercice 4
Ecrire un programme qui demande à l'utilisateur les coordonnées de 2 points distincts du plan et qui affiche les coordonnées du point milieu.

Exercice 5
Ecrire un programme qui demande à l'utilisateur une valeur pour U_0, r et n et qui affiche la $n^{ième}$ valeur de la suite arithmétique définie par U_0 et $U_{n+1} = U_n + r$. (On rappelle la propriété : $U_n = U_0 + n.r$).

STRUCTURES DE TRAITEMENT ALTERNATIVES ET ITÉRATIVES

Les instructions **structurées** sont des instructions composées d'autres instructions devant être exécutées sous certaines conditions (instructions conditionnelles) ou répétées plusieurs fois (instructions répétitives).

I- Instructions conditionnelles

Instruction IF…Then

Syntaxe : IF <Cond> THEN <Bloc> ;

Si <Cond> est vraie, le bloc d'instructions <Bloc> sera exécuté, sinon il sera ignoré.

Cette instruction représente l'alternative simple (instruction de choix unaire).

Exemple : if a=b then a :=a+b ;

Instruction IF…THEN…ELSE

Syntaxe : IF <Cond> THEN <Bloc1> ELSE <Bloc2>;

Si <Cond> est vraie, le bloc d'instructions <Bloc1> sera exécuté et le bloc d'instructions <Bloc2> sera ignoré, sinon c'est le bloc d'instructions <bloc2> qui sera exécuté et le bloc d'instructions <Bloc1> sera ignoré.

Cette instruction représente l'alternative complète (instruction de choix binaire).

Exemple : if a>b then a:=a-b

Else a:=b-a;

Remarque : la clause ELSE ne doit pas être précédée par un point virgule ' **;**'.

Instruction CASE

L'instruction CASE est constituée d'une expression de type scalaire, représentant le sélecteur, et d'une liste de blocs d'instructions ; chacun étant

précédé par une étiquette de cas de même type que le sélecteur. Le bloc d'instructions exécuté est celui dont l'étiquette de cas correspond à la valeur courante du sélecteur.

Une étiquette de cas est constituée de tous nombres de constantes ou d'intervalles, séparés par des virgules et terminés par le symbole ' :'.

Syntaxe :

```
CASE <Sélecteur> OF
        Val1 : <Bloc1> ;
        Val2 : <Bloc2> ;
        .....
        Valn : <Blocn> ;
        ELSE <Bloc>;
END;
```

II- Instructions répétitives

Structures	En langage Pascal
Tant que	While <expression> Do <bloc>
Répéter - jusqu'à	Repeat<instruction> Until <expression>
Pour	For comp :=Vint To Valf Do <bloc> For comp :=Vint DownTo Valf Do <bloc>

III- Enoncé des exercices

Exercice 6

Ecrire un programme qui échange les contenus de 2 données si elles sont de signes contraires.

Exercice 7

Ecrire un programme qui échange les contenus de 2 données si elles sont de signes contraires, sinon, il met leur produit dans la première donnée et leur somme dans la deuxième

Exercice 8

Ecrire un programme qui, étant donné un mois et son premier jour, affiche le premier jour du mois suivant.

Exercice 9

Ecrire un programme qui calcule la somme des N premiers termes positifs.

Exercice 10

Ecrire un programme qui calcule la somme des N premiers termes positifs impairs.

Exercice 11
Ecrire un programme qui calcule la somme des N premiers termes positifs pairs non multiples de 3.

Exercice 12
Ecrire un programme qui calcule la somme $1/2 + 1/4 + 1/8 +\ldots+1/2^n$ (n Є N*).

Exercice 13
Ecrire un programme qui calcule la somme $1+1/2-1/4+ 1/8-1/16+\ldots\pm1/2^n$ (n Є N*).

Exercice 14
Ecrire un programme qui donne le nombre N tel que la série 1+1/2-1/3+1/4-1/5+…±1/N donne un résultat égal (à 1/100 près) à 1,33.

Exercice 15
Ecrire un programme qui donne la plus grande valeur inférieure à 1000 de la somme $1+2+4+8+\ldots+2^N$.

Exercice 16
Ecrire un programme qui calcule la somme $1+x+x^2+\ldots+x^n$ (x réel et n entier).

Exercice 17
Calcul approché de $\sqrt{x}$
Soient les deux suites: $A_{n+1}= (A_n+G_n)/2$ et $G_{n+1}=2A_nG_n/(A_n+G_n)$ avec $A_0=x$ et $G_0=1$, on montre que $\lim_{n\to\infty} A_n=\lim_{n\to\infty} G_n=\sqrt{x}$
Ecrire un programme qui donne la valeur de $\sqrt{x}$ avec une précision relative Є=10^{-6}, c'est à dire |An-Gn|/|Gn|<Є

Exercice 18
Sachant que la somme 1-1/3+1/5-1/7+1/9-…, tend vers Π/4. Écrire un programme qui calcule le nombre Π à 10^{-6} près

Exercice 19
Le développement limité de la fonction sinus au voisinage de zéro est $\sin x= x - x^3/3! + x^5/5! -\ldots+(-1)^p x^{2p+1}/(2p+1)!+\ldots$
Ecrire un programme qui calcule sin x à 10^{-6} près.

Exercice 20
Ecrire un programme qui calcule la somme, le produit et la différence de deux données numériques.

Exercice 21
Ecrire un programme qui retourne le code d'une donnée de type ***Caractere***.

Exercice 22

Ecrire un programme qui calcule le salaire net d'un employé, sachant que celui-ci a assuré un certain nombres d'heures de travail à un prix fixe par heure, et que l'employeur doit réduire de son salaire des charges qui sont calculées avec un coefficient donné.

Exercice 23

Ecrire un programme qui permet de reprendre l'exercice précédant en considérant que l'employé a assuré des heures normales, des heures à 25% et des heures à 50%.

Exercice 24

Ecrire un programme qui retourne si une donnée numérique est paire ou impaire (utiliser le reste de la division par 2).

Exercice 25

Ecrire un programme qui permet de lire trois données numériques et retourne si la troisième donnée est la somme des deux autres ou pas.

Exercice 26

Ecrire un programme qui, à partir du salaire brut d'un employé, détermine l'impôt à payer sachant que les règles de calcul des impôts sont comme suit :

salaire brut(SB)	**l'impôt à payer**
SB<1500	0%
1500<=SB<3000	10% du SB
3000<=SB<5000	450+30%(SB-3000)
SB>=5000	750+40%(SB-5000)

Exercice 27

Ecrire un programme qui fournit les racines de l'équation $Ax^2+Bx+C=0$.

Exercice 28

Ecrire un programme qui, étant donnée une date (jour, mois, année), affiche la date du jour suivant.

Exercice 29

Ecrire un programme qui, étant donnée une date (jour, mois, année), affiche la date du jour précédent.

Exercice 30

Ecrire un programme qui calcule la somme 5+6+7+….+N (N>=5).

Exercice 31

Ecrire un programme qui calcule le produit des N (N>0) premiers entiers positifs.

Exercice 32

Ecrire un programme qui calcule la somme 1+1/2+1/4+1/6+…+1/2N (N>0).

Exercice 33

Ecrire un programme qui échange les contenus de trois données numériques si leur somme est paire, sinon il met la somme des trois dans la première donnée, leur produit dans la seconde et la valeur zéro dans la troisième.

Exercice 34

Ecrire un programme qui calcule la somme 1-1/2+1/3-1/4+…±1/N.

NOTION DE SOUS-PROGRAMMES

Un sous-programme permet :

- d'éviter d'appliquer à plusieurs endroits d'un programme le même traitement, même s'il porte sur des objets différents.
- d'appliquer sous forme transportable un traitement dont on prévoit que d'autres programmes peuvent aussi en avoir besoin.
- de dégager d'un programme des tâches de moindre importance ne laissant dans son corps que la partie importante de l'algorithme d'où le gain de lisibilité et de sécurité.

Les sous-programmes en Pascal sont des procédures et des fonctions.

I.1 Les procédures

Une procédure doit être déclarée dans un programme, ou dans une procédure, avant le corps du programme (de la procédure) et après les déclarations des variables.

Le nom de la procédure est un identificateur qui, comme les autres déclarations environnantes, sera local au programme ou à la procédure l'environnant.

La structure d'une procédure est à peu près celle d'un programme, l'en-tête diffère, la procédure se termine par un point virgule au lieu d'un point.

Syntaxe :
```
procedure <Ident> ;
        <Déclarations> ;
Begin
        <Instructions> ;
End ;
```

Paramètres formels

La déclaration des procédures peut être suivie d'une liste de paramètres entre parenthèses. Un paramètre formel est destiné à être remplacé, chaque fois que la procédure est activée, par un autre objet dit paramètre effectif, le plus souvent une variable, ou le résultat d'une expression, de la procédure appelante.

Syntaxe : Procedure <Nom> (<mode> <Ident>.., <Ident> : <Type> ;.. ; <mode> <Ident>,.., <Ident> : <Type>) ;

Où <mode> = rien ou var.

Paramètres par valeurs et paramètres par adresse

Les objets sont transmis comme données dont la valeur doit être **inchangée** par la procédure appelée (on dit qu'on a une lecture pure).

Les objets sont transmis comme résultat dont la valeur doit être calculée par la procédure appelée et transmise à la procédure appelante (on dit qu'on a une écriture pure).

Les objets sont transmis comme données qui doivent être modifiées par la procédure appelée (on dit qu'on a une lecture-écriture).

En Pascal, la transmission des paramètres se fait de deux manières :

- Transmission par valeur (lecture pure).
- Transmission par adresse (écriture pure, lecture-écriture).

On indique les paramètres transmis par adresse, précédés par le mot réservé Var, les autres ne sont précédés par aucune chose.

I.2 Les fonctions

Une fonction en Pascal se comporte comme une procédure sauf que la fonction doit fournir un résultat, elle a donc un type.

Syntaxe : Function <Ident> : <Type> ;
Function <Ident> (<liste_paramètres>) : <Type> ;

Dans le corps de la fonction, on doit trouver quelque part l'affectation d'une valeur à l'identificateur de la fonction.

Exemple :

```
Function Minimum (x, y : Integer ) : Integer ;
    Begin
        If x<y then Minimum :=x
        Else Minimum:=y
    End;
```

III. La récursivité

Un sous-programme est dit récursif s'il s'appelle lui-même. Il doit contenir une condition d'arrêt. Considérons l'exemple de la fonction pgcd(a, b) qui retourne le plus grand diviseur commun des deux entiers a et b :

En langage Pascal

```
Funtion pgcd (a,b : integer): integer;
Begin
   If (a=b) then pgcd:=a
   Else if (a>b) then pgcd:=pgcd(a-b,b)
        Else pgcd:=pgcd(a,b-a)
End;
```

IV- Enoncé des exercices

Exercice 35
Ecrire une fonction paramétrée qui retourne si un nombre donné est premier ou non.

Exercice 36
Ecrire une fonction paramétrée qui retourne si un nombre donné est parfait ou non.

Exercice 37
Ecrire une fonction paramétrée qui retourne si deux nombres donnés sont amis ou non.

Exercice 38
Ecrire une fonction paramétrée qui retourne l'inverse d'un nombre entier donné.

Exercice 39
Ecrire une fonction récursive permettant de calculer le PGDC de deux nombres entiers positifs A et B.

Exercice 40
Ecrire une fonction récursive permettant de calculer le PPMC de deux nombres entiers positifs A et B.

Exercice 41
Ecrire une fonction récursive qui permet de calculer le factoriel d'un nombre donné.

Exercice 42

Ecrire une fonction récursive qui permet de calculer la puissance d'un entier donné.

Exercice 43

Ecrire une fonction récursive qui calcule la valeur de la fonction d'Ackermann « A » définie pour m>0 et n>0 par :

A(m, n)=A((m-1),A(m,n-1)) pour n>0, m>0 ;
A(0,n)=n+1 pour n>0;
A(m,0)=A(m-1,1) pour m>0 ;

Exercice 44

Ecrire une fonction qui fournit le nombre de chiffres d'un entier donné.

Exercice 45

Ecrire une procédure qui permet de dessiner la lettre X, à l'aide d'espaces et d'une "lettre" fournie par l'utilisateur, auquel on demande aussi la "hauteur" du dessin qu'il désire obtenir.

Par exemple : avec les réponses a et 5, elle donnera :

```
a   a
 a a
  a
 a a
a   a
```

Exercice 46

Soit un programme qui visualise un menu composé de trois rubriques : Hors d'œuvre, plat chaud et dessert.

Ecrire une procédure qui permet d'effectuer, à chaque fois, un choix exclusif.

Exercice 47

Ecrire une procedure paramétrée, qui permet l'échange des contenus de deux paramètres formels par valeur A et B. Appeler cette procédure dans un programme principal.On écrira les résultats dans le corps de la procédure, et dans le programme principal.

Exercice 48

Ecrire une procedure paramétrée, qui permet l'échange des contenus de deux paramètres formels par adresse A et B. Appeler cette procédure dans un

programme principal.On écrira les résultats dans le corps de la procédure, et dans le programme principal.

Exercice 49

Ecrire une procédure qui affiche tous les nombres premiers qui sont compris entre 1 et 200.

Exercice 50

Ecrire une procédure qui affiche tous les nombres parfaits qui sont compris entre 1 et 200.

Exercice 51

Ecrire une procédure qui permet d'afficher tous les diviseurs d'un entier N ainsi que leur nombre.

Exercice 52

Affichez un triangle isocèle formé d'étoiles sur N lignes (N est fourni au clavier).

Exemple : N=8

```
        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***************
```

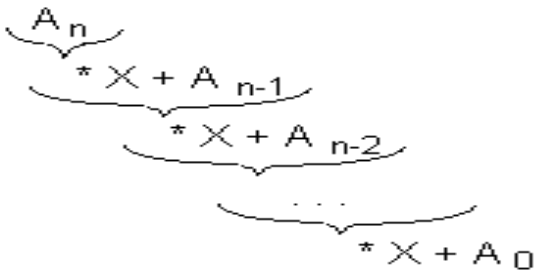
Exercice 53

Calculer pour une valeur X donnée du type **réel** la valeur numérique d'un polynôme de degré n:

$$P(X) = A_nX^n + A_{n-1}X^{n-1} + ... + A_1X + A_0$$

Les valeurs de n, des coefficients A_n, ..., A_0 et de X seront entrées au clavier.

Utiliser le *schéma de Horner* qui évite les opérations d'exponentiation lors du calcul:



Exercice 54

Ecrire une procédure qui affiche la table des produits pour N variant de 1 à 10 :

```
X*Y I   0   1   2   3   4   5   6   7   8   9  10
---------------------------------------------------
0  I   0   0   0   0   0   0   0   0   0   0   0
1  I   0   1   2   3   4   5   6   7   8   9  10
2  I   0   2   4   6   8  10  12  14  16  18  20
3  I   0   3   6   9  12  15  18  21  24  27  30
4  I   0   4   8  12  16  20  24  28  32  36  40
5  I   0   5  10  15  20  25  30  35  40  45  50
6  I   0   6  12  18  24  30  36  42  48  54  60
7  I   0   7  14  21  28  35  42  49  56  63  70
8  I   0   8  16  24  32  40  48  56  64  72  80
9  I   0   9  18  27  36  45  54  63  72  81  90
10 I   0  10  20  30  40  50  60  70  80  90 100
```

III- Enoncé des exercices

Exercice 55
Ecrire une procédure qui affiche les nombres négatifs d'une liste réelle.

Exercice 56
Ecrire une procédure qui met à zéro la diagonale d'une matrice carrée.

Exercice 57
Ecrire une procédure qui affiche l'occurrence d'existence d'un nombre réel dans une liste de nombres réels.

Exercice 58
Ecrire une procédure qui met le plus petit élément d'une liste au début de celle-ci.

Exercice 59
Ecrire une procédure qui met les éléments négatifs d'une liste à gauche et les éléments positifs à droite de la liste.

Exercice 60
Ecrire une procédure qui classe une liste de notes de la plus petite à la plus grande.

Exercice 61
Etant donné N étudiants, leurs notes correspondantes à M matières et leur moyenne. Ecrire une procédure qui affiche à côté de chaque étudiant son classement.

Exercice 62
Le tri à bulles est un tri par échange. Le principe de base est de réordonner les couples non classés tant qu'ils existent. La méthode de tri à bulles consiste à parcourir la liste en comparant deux éléments successifs et en les permutant s'il y a lieu. Ecrire une procédure qui réalise ce tri.

Exercice 63

Un palindrome est un mot, ou une phrase, lisible indifféremment de la gauche vers la droite ou inversement. Ecrire une fonction qui retourne si une chaîne de caractères est un palindrome ou non.

Exercice 64

Ecrire une fonction qui retourne la fréquence d'occurenced'un mot dans une phrase.

Exercice 65

Ecrire une procédure qui enlève tous les blancs au début d'une phrase et qui laisse un seul blanc entre les différents mots de la phrase.

Exercice 66

Soit un nombre entier positif N. Ecrire une fonction qui donne son équivalent dans une base donnée B ($2 \leq B \leq 16$).

Exercice 67

Soit N un nombre donné en base B ($B \neq 10$). Ecrire une fonction qui donne son équivalent en base 10.

Exercice 68

Soit N un nombre donné en base B1. Ecrire une fonction qui donne son équivalent en base B2.

Exercice 69

Si on est amené à réaliser des opérations sur des nombres entiers très grands, on peut utiliser les chaînes de caractères pour représenter ces nombres et ensuite faire l'opération par bloc.
Ecrire une procédure pour réaliser l'opération d'addition sur de tels nombres.

Exercice 70

Ecrire une procédure qui permet de passer de la représentation d'un nombre en chiffres romains à sa représentation décimale.

Exercice 71

Ecrire une procédure qui permet de passer de la représentation décimale d'un nombre à sa représentation en chiffres romains.

Exercice 72

Ecrire une fonction qui retourne le déterminant d'une matrice carrée.

Exercice 73

Ecrire une procédure qui calcule l'inverse d'une matrice carrée.

Exercice 74

Un carré magique est un carré divisé en cellules dans lesquelles les nombres entiers, à partir de 1 sont disposés de telle sorte que les sommes de chaque ligne, de chaque colonne et de chaque diagonale soient égales.

Exemple :

6	1	8
7	5	3
2	6	4

Ecrire une procédure qui permet de réaliser le carré magique d'ordre n (n impair).

Exercice 75

Ecrire un programme qui permet de saisir et d'afficher les informations d'une liste d'étudiants.

Exercice 76

Ecrire les procédures qui donnent le calcul sur les nombres complexes au moyen d'un type enregistrement.

SOLUTIONS DES EXERCICES

Pascal
Exercice 1 Ecrire un programme qui permet d'afficher le message suivant : mon premier programme.
Program afficher ; Begin Writeln('mon premier programme') ; Readln ; End .
Exercice 2 Ecrire un programme qui demande à l'utilisateur les valeurs de 2 entiers x et y, qui permute leurs valeurs et qui les affiche.

Program Echange;
var x,y,tmp:integer;
begin
write('donner deux entiers differents:');
readln(x,y);
writeln('x=',x,' et y=' ,y);
tmp:=x;
x:=y;
y:=tmp;
writeln('x=',x,' et y=',y);
readln;
end.

Exercice 3 Ecrire un programme qui échange 3 entiers.

Program Echange;
var x,y,tmp,z:integer;
begin
write('donner trois entiers differents:');
readln(x,y,z);
writeln('x=',x,' et y=' ,y,'et z=',z);
tmp:=x;
x:=y;

```
y :=z ;
    z:=tmp;
    writeln('x=',x,' et y=',y,' et z=',z);
    readln;
end.
```

Exercice 4

Ecrire un programme qui demande à l'utilisateur les coordonnées de 2 points distincts du plan et qui affiche les coordonnées du point milieu.

```
program milieu ;
var     x1, y1, x2, y2, xm, ym : real  ;
begin
        writeln ('Entrez les coordonnées du premier point') ; readln ( x1 ) ; readln (
y1 );
        writeln ('Entrez les coordonnées du deuxième point') ; readln ( x2 ) ; readln (
y2 );
        xm := (x1 + x2) / 2 ;  ym := (y1 + y2) / 2 ;
        writeln ('Les coordonnées du point milieu sont :', xm :5:2, ym:5:2) ;
end.
```

Exercice 5

Ecrire un programme qui demande à l'utilisateur une valeur pour U0, r et n et qui affiche la nième valeur de la suite arithmétique définie par U0 et Un+1 = Un + r. (On rappelle la propriété : Un = U0 + n.r).

```
program suite ;
var U0, Un, r : real  ;        n : integer ;
begin
        writeln ('Entrez les valeurs de U0, r et n') ;
          readln ( U0,r,  n );
        Un := U0 + n*r;
         writeln ('La ',n:3,'ième valeur de la suite est :', Un :5:2) ;
end.
```

Exercice 6

Ecrire un programme qui échange les contenus de 2 données si elles sont de signes contraires.

```
program signe_contraire;
var tmp,x,y:integer;
begin
    write('donner deux entiers differents:');
    readln(x,y);
    writeln('x=',x,' et y=' ,y);
```

```
    if x*y <0 then
    begin
        tmp:=x;
        x:=y;
        y:=tmp;
        writeln('les deux entiers sont de signes contraires donc apres echange');
        writeln('x=',x,' et y=',y);
    end
    else
        writeln('les deux entiers sont de meme signes donc pas d"echange');
    readln;
end.
```

Exercice 7

Ecrire un programme qui échange les contenus de 2 données si elle sont de signes contraires, sinon, il met leur produit dans le premier et la somme dans le deuxième

```
program somme_produit;
var tmp,x,y:integer;
begin
write('donner deux entiers:');
    readln(x,y);
    writeln('x=',x,' et y=' ,y);
    if x*y<0 then
    begin
        tmp:=x;
        x:=y;
        y:=tmp;
    end
    else
    begin
       tmp:=x;
       x:=y*x;
       y:=y+tmp;
    end;
    writeln('apres operation');
    writeln('x=',x,' et y=',y);
    readln;
end.
```

Exercice 8

Ecrire un programme qui étant donnée un mois et son premier jour, affiche le premier jour du mois suivant

```
program premier_jour;
var nbj,annee,mois,premj:integer;
begin
    write('entrer une annee de votre choix:');
    readln(annee);
    write('entrer un mois et son premier jour:');
    readln(mois,premj);
    case mois of
        1,3,5,7,8,10,12: nbj:=31;
        4,6,9,11: nbj:=30;
        2: begin
          if annee mod 100 = 0 then annee:=annee div 100;
          if annee mod 4 = 0 then nbj:=29
          else nbj :=28;
          end;
    end;
    premj:=(premj+nbj) mod 7;
    writeln('le premier jour du mois suivant est:',premj);

    readln;
end.
```

Exercice 9

Ecrire un programme qui calcule la somme des N premiers termes positifs

```
program somme;
var N,i,s:integer;
begin
    write('entrer un nombre positif:');
    readln(N);
    s:=0;
    if N>0 then
      for i:=1 to N do
          s:=s+i;
    writeln('la somme de N premiers termes positifs:',N);;
    readln
end.
```

Exercice 10

Ecrire un programme qui calcule la somme des N premiers termes positifs impaires

```
program som_impaire;
var N,i,s:integer;
begin
    write('entrer un entier positif:');
    readln(N);
    i:=1;s:=0;
    while(i<2*N) do
    begin
        s:=s+i;
        i:=i+2;
    end;
    writeln('somme de ',N,' premiers termes positifs impaires est: ',s);
    readln;
end.
```

Exercice 11

Ecrire un programme qui calcule la somme des N premiers termes positifs pairs nom multiple de 3

```
program som_paires;
var N,i,s,cp:integer;
begin
    write('entrer un entier positif:');
    readln(N);
    i:=0;s:=0;cp:=0;
    while(cp<N) do
    begin
        i:=i+2;
        if (i mod 3 <> 0) then
          begin
              s:=s+i;
              cp:=cp+1;
          end;
    end;
    writeln('somme de ',N,' premiers termes positifs non multiples de 3 est: ',s);
    readln;
end.
```

Exercice 12

Ecrire un programme qui calcule la somme $1/2 + 1/4 + 1/8 + \ldots + 1/2^n$ ($n \in N^*$)

```
program calcul;
var n,i:integer;
   s,u:real;
begin
s:=0;
u:=1/2;
i:=1;
write('donner un entier non nul: ');readln(n);
while i<=n do
     begin
        s:=s+u;
        u:=u/2;
        i:=i+1;
     end;
writeln('la somme est: ',s:10:5);
readln;
end.
```

Exercice 13

Ecrire un programme qui calcule la somme $1+1/2-1/4+ 1/8-1/16+\ldots\pm1/2^n$ ($n \in N^*$)

```
program calcul;
var n,i:integer;
   s,u:real;
begin
write('donner un entier non nul: ');readln(n);
i:=1;
S:=1;
u:=1/2;
while i<=n do
     begin
     if i mod 2=0 then
        s:=s-u
     else s:=s+u;
        u:=u/2;
        i:=i+1;

     end;
writeln('la somme est: ',s:10:5);
readln;
```

```
end.
```

Exercice 14

Ecrire un programme qui donne le nombre N tel que la série 1+1/2-1/3+1/4-1/5+…±1/N donne un résultat égal (à 1/100 près) à 1,33

```
program calcul;
var S:real;
   i:longint;
begin
    S:=1;
    i:=1;
    while abs(S - 1.33) > 0.001 do
    begin
        if (i mod 2=0) then  S:=S-(1/(i+1))
        else S:=S+(1/(i+1));
        i:=i+1;
    end;
    write('le nombre tel que la serie donne un resultat = 1,33 est:',s:10:5) ;
    readln;
end.
```

Exercice 15

Ecrire un programme qui donne la plus grande valeur inférieure à 1000 de la somme 1+2+4+8+…+2N

```
program calcul;
var u,s:integer;
begin
u:=2;
s:=1;
while s<1000 do
     begin
         s:=s+u;
         u:=u*2;
     end;
     s:=s-(u div 2);
writeln('la plus grande valeur inferieur a 1000 de la somme est: ',s);
readln;
end.
```

Exercice 16

Ecrire un programme qui calcule la somme $1+x+x^2+\ldots+x^n$ (x réel et n entier).

```
program calcul;
var x,s,u:real;
   n,i:integer;
begin
write('donner un reel et un entier:');
readln(x,n);
i:=1;
s:=1;
u:=x;
while i<=n do
     begin
          s:=s+u;
          u:=u*x;
          i:=i+1;
     end;
writeln('la somme est: ',s:10:5);
readln;
end.
```

Exercice 17

Calcul approché de $\sqrt{x}$

Soient les deux suites: $A_{n+1}= (A_n+G_n)/2$ et $G_{n+1}=2A_nG_n/(A_n+G_n)$ avec $A_0=x$ et $G_0=1$, on montre que $\lim A_n=\lim G_n=\sqrt{x}$

$n\rightarrow\infty$ $n\rightarrow\infty$

Ecrire un programme qui donne la valeur de $\sqrt{x}$ avec une précision relative Є=10^{-6}, c'est à dire |An-Gn|/|Gn|<Є

```
program racine_carre;
var x,A,G,B:real;
begin
    write('donner un nombre non nul: ');
    readln(x);
    A:=x;
    G:=1;
    while abs((A-G)/G)>0.000001 do
    begin
         B:=A;
         A:=(A+G)/2;
         G:=2*B*G/(B+G);
```

```
    end;
    writeln('la racine de ',x:10:5,' est: ',A:10:5);
    readln;
end.
```

Exercice 18

Sachant que la somme 1-1/3+1/5-1/7+1/9... tend vers ∏/4. écrire un programme donnant le nombre ∏ à 10^{-6} près

```
program pi;
var s,t:real;
   i:integer;
begin
    s:=4;
    i:=1;
    t:=1/3;
    while t>0.000001 do
    begin
        t:=4/(2*i+1);
        if i mod 2=0 then
          s:=s+t
        else s:=s-t;
        i:=i+1;
    end;
    writeln('pi =',s:10:5);
    readln;
end.
```

Exercice 19

Le développement limité de la fonction sinus au voisinage de zéro est : $\sin x = x - x^3/3! + x^5/5! - \ldots + (-1)^p x^{2p+1}/(2p+1)! + \ldots$

Ecrire un programme qui calcule sin x à 10^{-6} près.

```
program sinx;
var s,x,u,t:real;
   v,i:integer;
begin
write('donner un nombre:');readln(x);
s:=0;
u:=x;
v:=1;
i:=1;
```

```
t:=u/v;
while t>0.000001 do
     begin
          if i mod 2=0 then s:=s-(u/v)
          else s:=s+(u/v);
          u:=u*x*x;
          v:=(2*i+1)*2*i*v;
          i:=i+1;
          t:=u/v;
     end;
     writeln('le sinus de ',x:10:5,'est: ',s:10:5);
     readln;
end.
```

Exercice 20

Ecrire un programme qui calcule la somme, le produit et la différence de deux données numériques.

```
program operation;
var x,y:integer;
   op:char;
begin
writeln('donner deux nombres x et y');
readln(x,y);
writeln('donner un op,rateur');
readln(op);
case(op) of
'/': begin
     if y<>0 then writeln('la division de x par y est ,gale …:',x/y)
     else  writeln(' division par z,ro');
      end;
'*':writeln('x*y=',x*y);
'+':writeln('x+y=',x+y);
'-':writeln('x-y=',x-y);
end;
readln;
end.
```

Exercice 21

écrire un programme qui retourne le code d'une donnée de type CARACTERE.

```
program code_ASCII;
var c:char;
asci:integer;
begin
writeln('donner un caractère:');
readln(c);
asci:=ord(c);
writeln('le code ASCII de ce caractère est:',asci);
readln;
end.
```

Exercice 22

Ecrire un programme qui calcule le salaire net d'un employé, sachant que celui-ci a assuré un certain nombres d'heures de travail à un prix fixe par heure, et que l'employeur doit réduire de son salaire des charges qui sont calculées avec un coefficient donné.

```
program salaire_net_employe;
var charges,sb,sn,coef,ph:real;
   nh:integer;
begin
writeln('donner le nombre d''heures de travail de cet employ,:');
readln(nh);
writeln('donner le prix d''une heure:');
readln(ph);
writeln('donner le coefficient:');
readln(coef);
sb:=nh*ph;
charges:=sb*coef;
sn:=sb-charges;
writeln('le salaire net de cet employ, est:',sn);
readln;
end.
```

Exercice 23

Ecrire un programme qui permet de reprendre l'exercice précédant en considérant que l'employé a assuré des heures normales, des heures à 25% et des heures à 50%.

```
program salaire_net_supplements;
var charges,sb,sn,coef,ph:real;
   nh25,nh50,nhn:integer;
begin
writeln('donner le nombre d''heures … 25% et celui … 50% ainsi que le nombre
d''heure normales  :');
```

```
readln(nh25,nh50,nhn);
writeln('donner le prix d''une heure:');
readln(ph);
writeln('donner le coefficient:');
readln(coef);
sb:=nhn*ph+nh25*ph*125/100+nh50*150/100*ph;
charges:=sb*coef;
sn:=sb-charges;
writeln('le salaire net de cet employ, est:',sn);
readln;
end.
```

Exercice 24

Ecrire un programme qui retourne si une donnée numérique est paire ou impaire (utiliser le reste de la division par 2).

```
program est_pair;
var n:integer;
bp:boolean;
begin
writeln('donner un entier positif:');
readln(n);
bp:=(n mod 2)=0;
if bp then writeln('cet entier est pair')
else writeln('cet entier n''est pas pair');
readln;
end.
```

Exercice 25

Ecrire un programme qui permet de lire trois données numériques et retourne si la troisième donnée est la somme des deux autres ou pas.

```
program est_somme;
var x,y,z:integer;
   bs:boolean;
begin
writeln('donner trois nombres:');
readln(x,y,z);
bs:=(x+y)=z;
if bs then writeln('la trosième donnée est égale à la somme des autres ')
else
  writeln('la trosième donnée n'' est pas égale à la somme des autres ');
readln;
end.
```

Exercice 26

Ecrire un programme qui, à partir du salaire brut d'un employé, détermine l'impôt à payer sachant que les règles de calcul des impôts sont comme suit :

salaire brut(SB)	l'impôt à payer
SB<1500	0%
1500<=SB<3000	10% du SB
3000<=SB<5000	450+30%(SB-3000)
SB>=5000	750+40%(SB-5000)

```
program impots;
var sb,impot:real;
begin
      writeln('donner le salaire brut de cet employé:');
      readln(sb);
if sb<1500 then writeln('l''impot à payer est de 0%')
else
   begin
   if sb<3000 then writeln('l''impot à payer est de:',sb*0,1)
   else
      begin
      if sb<5000 then writeln('l''impot à payer est de:',450+30/100*(sb-3000))
      else
      writeln('l''impot à payer est de:',750+40/100*(sb-5000));
      end;
   end;
readln
   end.
```

Exercice 27

Ecrire un programme qui fournit les racines de l'équation $Ax^2+Bx+C=0$.

```
program eqation ;
var a,b,c:integer;
   delta:real;
begin
writeln('donner a, b et c');
readln(a,b,c);
if a=0 then
     begin
     if b=0 then begin
              if c=0 then writeln('S=R')
```

```
            else writeln('S=VIDE');
            end
      else writeln('la solution est:',-c/b);
      end
else begin
    delta:=b*b-4*a*c;
    if delta>=0 then writeln('les racines de cette équation sont: x1=',(-b-
sqrt(delta))/(2*a),'x2=', (-b+sqrt(delta))/(2*a))
    else writeln('pas de solution');
    end;
readln;
end.
```

Exercice 28

Ecrire un programme qui, étant donnée une date (jour, mois, année), affiche la date du jour suivant.

```
program jour_suivant;
var j,m,a,ms,js,as:integer;
   biss,dj:boolean;
begin
writeln('donner la date d"aujourd"hui:');
readln(j,m,a);
if( a mod 100)=0 then biss:=(a mod 400)=0
else biss:=(a mod 4)=0;
case m of
       1,3,7,8,10,12:dj:=(j=31);
       4,6,9,11:dj:=(j=30);
       2:if biss then dj:=(j=29)
      else  dj:=(j=28);
 end;
 if dj then begin
          js:=1;
          if m=12 then begin
                    ms:=1;
                    as:=a+1;
                    end
          else begin
              ms:=m+1;
              as:=a;
              end;
              end
```

```
 else begin
     js:=j+1;
     ms:=m;
     as:=a;
     end;
writeln('le jour suivant est :',js,'/',ms,'/',as);
readln;
end.
```

Exercice 29

Ecrire un programme qui, étant donnée une date (jour, mois, année), affiche la date du jour précédant.

```
program jour_precedant;
var nj,j,m,a,mp,jp,ap:integer;
   biss:boolean;
begin
writeln('donner la date d"aujourd"hui:');
readln(j,m,a);
if( a mod 100)=0 then biss:=(a mod 400)=0
else biss:=(a mod 4)=0;
if m=1 then m:=13;
case m-1 of
        1,3,7,8,10,12:nj:=31;
        4,6,9,11:nj:=30;
        2:if biss then nj:=29
      else  nj:=28;
 end;
 if j=1 then begin
           if m=1 then begin
                     ap:=a-1;
                     jp:=nj;
                     mp:=12;

                     end
           else begin
               mp:=m-1;
               ap:=a;
               jp:=nj;
               end;
               end
 else begin
```

```
    jp:=j-1;
    mp:=m;
    ap:=a;
    end;
writeln('le jour précédant est :',jp,'/',mp,'/',ap);
readln;
end.
```

Exercice 30

Ecrire un programme qui calcule la somme 5+6+7+….+N (N>=5).

```
Program Somme_entiers;
var N,i,S:integer;
begin
writeln('donner un entier >=5:');
readln(N);
S:=0;i:=5;
while i<=N do
begin
S:=S+i;
i:=i+1;
end;
writeln('la somme est égale à',S);
readln;
end.
```

Exercice 31

Ecrire un programme qui calcule le produit des N (N>0) premiers entiers positifs.

```
program Produit_Entiers;
var P,i,N:integer;
begin
writeln('donner un entier >0:');
readln(N);
if N<=0 then writeln('il faut donner un entier > 0!')
else
P:=1;
for i:=1 to N do
P:=P*i;
writeln('le produit des N premiers entiers est :',P);
end.
```

Exercice 32

Ecrire un programme qui calcule la somme 1+1/2+1/4+1/6+…1/2N (N>0).

```
program Serie_Fraction;
```

```
var i,N:integer;
S:real;
begin
writeln('donner un entier N>0');
readln(N);
S:=1;
i:=1;
repeat
     begin
     S:=S+(1/(2*i));
     i:=i+1;
     end;
until i=N+1;
writeln('S=,'S);
readln;
end.
```

Exercice 33

Ecrire un programme qui échange les contenus de trois données numériques si leur somme est paire, sinon il met la somme des trois dans la première donnée, leur produit dans la seconde et la valeur zéro dans la troisième.

```
program echange_cond_trois;
uses crt;
var s,tmp,x,y,z:integer;
begin
clrscr;
writeln('donner trois entiers:');
readln(x,y,z);
s:=x+y+z;
if(s mod 2)=0 then begin
                tmp:=x;
                x:=y;
                y:=z;
                z:=tmp;
                writeln('x=',x,'  ','y=',y,'  ','z=',z);
                end

else begin
    tmp:=x;
    x:=x+y+z;
    y:=y*tmp*z;
```

```
    z:=0;
    writeln('x=',x,'  ','y=',y,'  ','z=',z);
    end;
readln;
end.
```

Exercice 34

Ecrire un programme qui calcule la somme :1-1/2+1/3-1/4+…+/-1/N

```
 program Serie_Alternative;
var i,N:integer;
   S:real;
   b:boolean;
begin
writeln('donner un entier >0');
readln(N);
S:=1;
for i:=2 to N do
     begin
     b:=(i mod 2)=0;
     if b then S:=S-1/i
     else S:=S+1/i;
     end;
writeln('S=',S);
readln;
end.
```

Les fonctions

Exercice 35

Ecrire une fonction paramétrée qui retourne si un nombre donné est premier ou non.

```
function est_premier(N:integer):boolean;
var i,nb:integer;
begin
nb:=0;
for i:=1 to N do
if N mod i =o then nb:=nb+1;
est_premier:=(nb=2);
end;
```

Exercice 36

Ecrire une fonction paramétrée qui rend si un nombre donné est parfait ou non.

```
function est_parfait(N:integer):boolean;
```

```
var  S,i:integer;
begin
S:=0;
for i:=1 to N-1 do
if N mod i = 0 then S:=S+i
est_parfait:=(N=S);
end;
```

Exercice 37

Ecrire une fonction paramétré qui rend si deux nombres données sont amis ou non.

```
function Amis(N:integer;M:integer):boolean;
var S,R,i,j:integer;
begin
S:=0;R:=0;
for i:=1 to N-1 do
if N mod i = 0 then S:=S+i;
for j:=1 toM-1 do
if M mod j = 0 then R:=R+j;
Amis:=((M=S) and (N=R));
end;
```

Exercice 38

Ecrire une fonction paramétrée qui retourne l'inverse d'un nombre entier donné

```
function inverse(n:longint):longint;
var r,inv:longint;
begin
    inv:=0;
    while n mod 10 =0 do n:=n div 10;
    while n mod 10 <>0 do
    begin
        r:=n mod 10;
        inv:=(inv*10)+r;
        n:=n div 10;
    end;
    inverse:=inv;
end;
```

Exercice 39

Ecrire une fonction récursive permettant de calculer le PGDC de deux nombres entiers positifs A et B.

```
function pgdc(a,b:integer):integer;
begin
```

```
if a>b then pgdc:=pgdc(a-b,b)
else if a<b then pgdc:=pgdc(a,b-a)
else pgdc:=a;
end;
```

Exercice 40

Ecrire une fonction récursive permettant de calculer le PPMC de deux nombres entiers positifs A et B.

```
function calcul_ppmc(A:integer,B:integer):integer;
var multiple,k:integer;
begin
if A<B then ppmc:=ppmc(B,A)
else
   begin
   multiple:=A;
   k:=2;
   end
while multiple mod B<> 0 do
   begin
   multiple:= A*k;
   k:=k+1;
   end;

ppmc:=multiple;
   end;
end;
```

Exercice 41

Ecrire une fonction récursive qui permet de calculer le factoriel d'un nombre donné.

```
function Factoriel(N:integer):integer;
begin
if N=0 then factoriel:=1
else
factoriel:= N*factoriel(N-1);
end;
```

Exercice 42

Ecrire une fonction récursive qui permet de calculer la puissance d'un entier donné.

```
function puissance(N:integer,k:integer):longint;
begin
if k=0 then puissance:=1
else
```

```
puissance:=N*puissance(N,k-1);
end;
```

Exercice 43

Ecrire une fonction récursive qui calcule la valeur de la fonction d'Ackermann « A » définie pour m>0 et n>0 par :

A(m, n)=A((m-1),A(m,n-1)) pour n>0, m>0 ;
A(0,n)=n+1 pour n>0;
A(m,0)=A(m-1,1) pour m>0 ;

```
function acker(n,m:integer):integer;
begin
if (m<0) or (n<0) then acker:=0
else if n=0 then acker:=m+1
     else if m=0 then acker:= acker(n-1,1)
          else acker:=acker(n-1,acker(n,m-1))
end;
```

Exercice 44

Ecrire une fonction qui fournit le nombre de chiffres d'un entier donné.

```
function NCHIFFRES (N:longint):integer;
var I:integer;
begin
 If N<0 then N:=N*(-1);
 I:=1;
 while N>10 do
  begin
    N:=N div 10;
    I:=I+1;
  end;
 NCHIFFRES:=I;
End;
```

Les procédures

Exercice 45

Ecrire une procédure qui permet de dessiner la lettre X, à l'aide d'espaces et d'une "lettre" fournie par l'utilisateur, auquel on demande aussi la "hauteur" du dessin qu'il désire obtenir. Avec les réponses : a et 5, elle donnera :

```
a   a
 a a
  a
 a a
a   a
```

```
procedure DessX(hauteur:integer;lettre:char);

var no_ligne,i:integer;
begin
 for no_ligne:=1 to ((hauteur+1) div 2)-1 do
 begin
  for i:=1 to no_ligne-1 do
    write(' ');
  write(lettre);
  for i:=0 to (hauteur-2*no_ligne)-1  do
   write(' ');
  write(lettre);
  writeln;
 end;
for i:=1 to ((hauteur+1) div 2)-1 do
  write(' ');
 write(lettre);
 writeln;
 for no_ligne:=1 to ((hauteur+1) div 2)-1 do
  begin
   for i:=0 to ((hauteur-2*no_ligne-1) div 2)-1 do
    write(' ');
  write(lettre);
  for i:=0 to (2*no_ligne-1)-1 do
   write(' ');
  write(lettre);
  writeln;
 end;
end;
```

Exercice 46

Soit un programme qui visualise un menu composé de trois rubriques : Hors d'œuvre, plat chaud et dessert.
Ecrire une procédure qui permet d'effectuer, à chaque fois, un choix exclusif.

```
program menu;
uses crt;
procedure Plat(choix1,choix2,choix3:string);
var choix:char;
begin
    choix:=readkey;
    writeln;
```

```
    write('-',choix,'-');
    if choix=choix1[1] then writeln(choix1)
    else if choix=choix2[1] then writeln(choix2)
    else if choix=choix3[1] then writeln(choix3);
end;
begin
    clrscr;
    plat('soupe','crudit','oeuf Dur');
    plat('steak','poulet','hamburger');
    plat('gateau','poire','fromage');
    readln;
end.
```

Exercice 47

Ecrire une procedure paramétrée, qui permet l'échange des contenus de deux paramètres formels par valeur A et B. Appeler cette procédure dans un programme principal.On écrira les résultats dans le corps de la procédure, et dans le programme principal.

```
program exchange;
var A:real;
   B: real;
procedure echange(x,y:real);
var temp:real;
begin
    writeln('Ecange:',x:10:2,y:10:2);
    temp:=x;
    x:=y;
    y:=temp;
    writeln('Echange:',x:10:2,y:10:2);
end;
begin
    readln(A,B);
    echange(A,B);
    writeln('Echange:',x:10:2,y:10:2);
    readln;
end.
```

Exercice 48

Ecrire une procedure paramétrée, qui permet l'échange des contenus de deux paramètres formels par adresse A et B. Appeler cette procédure dans un programme principal.On écrira les résultats dans le corps de la procédure, et dans le

programme principal.

```
program exchange;
var A:real;
   B:real;
procedure echange(var x,y:real);
var temp:real;
begin
    writeln('Ecange:',x:10:2,y:10:2);
    temp:=x;
    x:=y;
    y:=temp;
    writeln('Echange:',x:10:2,y:10:2);
end;
begin
    readln(A,B);
    echange(A,B);
    writeln('Echange:',x:10:2,y:10:2);
    readln;
end.
```

Exercice 49

Ecrire une procédure qui affiche tous les nombres premiers qui sont compris entre 1 et 200.

```
procedure Nombres_premiers;
 var  i,j,nb:integer;
 begin
 for j:=1 to 200  do
 begin
 nb:=0;
 for i:=1 to (j div 2) do
 if( j mod i =0) then nb:=nb+1;
 if nb=1 then writeln(j);
 end;
 end;
```

Exercice 50

Ecrire une procédure qui affiche tous les nombres parfaits qui sont compris entre 1 et 200.

```
procedure Nombres_parfaits;
 var  i,j,nb,S:integer;
 begin
```

```
 for j:=1 to 200  do
 begin
 S:=0;
 for i:=1 to j-1 do

 if( j mod i =0) then S:=S+I;
 if S=j then writeln(j);
 end;
 end;
```

Exercice 51

Ecrire une procédure qui permet d'afficher tous les diviseurs d'un entier N ainsi que leur nombre.

```
procedure Diviseurs(N :integer) ;
var i,nb:integer;
begin
nb:=0;
writeln('Les diviseurs de ',N,'sont :') ;
for i:=1 to N do
if N mod i =0 then
    begin
    writeln(i);
    nb:=nb+1;
    end;
writeln('Le nombre de diviseurs de ',N,'est :',nb);
end ;
```

Exercice 52

Affichez un triangle isocèle formé d'étoiles de N lignes (N est fourni au clavier):

Nombre de lignes : 8

```
                *
               ***
              *****
             *******
            *********
           ***********
          *************
         ***************
```

```
procedure DessinerTri(L:integer);
var
  k:integer; {compteur des lignes}
  ESP,I,j:integer;
```

```
begin
for k:=1 to L do
begin
 ESP := L-k;
 for I:=1 to ESP do
    write(' ');
 for j:=1 to 2*k-1 do
 write('*');
 writeln;
 end;
 readln;
 end ;
```

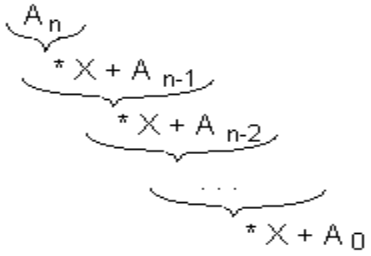
Exercice 53

Calculer pour une valeur X donnée du type float la valeur numérique d'un polynôme de degré n:

$P(X) = A_nX^n + A_{n-1}X^{n-1} + ... + A_1X + A_0$

Les valeurs de n, des coefficients A_n, ..., A_0 et de X seront entrées au clavier.

Utiliser le schéma de Horner qui évite les opérations d'exponentiation lors du calcul:




```
procedure polynome(N:integer;X:real);
var    I:integer;
       A,B:real;
begin
for I:=1 to N+1 do
begin
 writeln('donner le ',I,'eme coefficient');
 readln(B);
 A:=A*X+B;
 end;
 writeln('Valeur du polynôme pour X = ',X,'est :',A);
 readln; end;
```

Exercice 54

Ecrire une procédure qui affiche la table des produits pour N variant de 1 à 10 :

```
X*Y I   0   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
0  I   0   0   0   0   0   0   0   0   0   0   0
1  I   0   1   2   3   4   5   6   7   8   9  10
2  I   0   2   4   6   8  10  12  14  16  18  20
3  I   0   3   6   9  12  15  18  21  24  27  30
4  I   0   4   8  12  16  20  24  28  32  36  40
5  I   0   5  10  15  20  25  30  35  40  45  50
6  I   0   6  12  18  24  30  36  42  48  54  60
7  I   0   7  14  21  28  35  42  49  56  63  70
8  I   0   8  16  24  32  40  48  56  64  72  80
9  I   0   9  18  27  36  45  54  63  72  81  90
10  I  0  10  20  30  40  50  60  70  80  90 100
```

```
program  tabMul;
 CONST MAX= 10;
  var I,J:integer;
 begin
 write('    X*Y I');
 for I:=0 to MAX do
 write(I:4);
 writeln;
 writeln('-----------------------------------------------');
 for J:=0 to max do
 begin
 write(J:7,' I');
 for I:=0 to Max do
 write(I*J:4);
 writeln;
 end;
 readln;
 end.
```

Les Tableaux

Exercice 55

Ecrire une procédure qui affiche les nombres négatifs d'une liste réelle

```
Type tab=array[0..50] of real ;
        Procedure nbre_negatif(a:tab;n:integer);
        Var i:integer;
```

```
        Begin
          for i:=0 to n-1 do
            if a[i]<0 then writeln(a[i]);
        End;
```

Exercice 56

Ecrire une procédure qui met à zéro la diagonale d'une matrice carrée.

```
type matrice=array[1..50,1..50] of integer;
procedure mise_zero(var a:matrice;n:integer);
var i:integer;
begin
    for i:=1 to n do
     a[i,i]:=0;
end;
```

Exercice 57

Ecrire une procédure qui affiche l'occurrence d'existence d'un nombre réel dans une liste de nombres réels

```
Type tab=array[0..50] of real ;
Procedure occurrence(a:tab;n:integer;nbre:real);
Var i,cpt:integer;
Begin
  cpt:=0;
  for i:=0 to n-1 do
    if a[i]=nbre then cpt:=cpt+1;
  writeln('L''occurrence d''existence du nombre ',nbre,' est ',cpt) ;
End;
```

Exercice 58

Ecrire une procédure qui met le plus petit élément d'une liste au début de celle-ci

```
type tab=Array[0..50] of real;
procedure min_debut(var a:tab;n:integer);
var i,ind:integer;
     tmp: real;
begin
  ind:=0;
  for i:=0 to n-1 do
    if a[i]<a[ind] then ind:=i;
  tmp:=a[0];
  a[0]:=a[ind];
  a[ind]:=tmp;
end;
```

Exercice 59

Ecrire une procédure qui met les éléments négatifs d'une liste à gauche et les éléments positifs à droite de la liste

```
type tab=Array[0..50] of real;
procedure negatif_gauche(var a:tab;n:integer);
var i,j:integer;
    tmp: real;
begin
  j:=0;
  for i:=0 to n-1 do
     if a[i]<0 then
     begin
        tmp:=a[j];
        a[j]:=a[i];
        a[i]:=tmp;
        j:=j+1;
     end;
 end;
```

Exercice 60

Ecrire une procédure qui classe une liste de notes de la plus petite à plus grande

```
type tab=Array[0..50] of real;
procedure tri(var note:tab;n:integer);
var i,j,min:integer;
    tmp:real;
begin
 for i:=0 to n-2 do
    begin
       min:=i;
       for j:=i+1 to n-1 do
          if note[j]<note[min] then min:=j;
        if min<>i then
          begin
                tmp:=note[i];
                note[i]:=note[min];
                note[min]:=tmp;
          end;
    end;
end;
```

Exercice 61

Etant donné N étudiants, leurs notes correspondantes à M matières et leur

moyenne. Ecrire une procédure qui affiche à côté de chaque étudiant son classement.

```
type matrice=array[1..50,1..50]of real;
procedure moyenne(var m:matrice;tl,tc:integer);
var i,j:integer;
   s:real;
begin
    for i:=1 to tl do
    begin
        s:=0;
        for j:=1 to tc-2 do
          s:=s+m[i,j];
        m[i,tc-1]:=s/(tc-2);
    end;
end;

procedure classement(var m:matrice;tl,tc:integer);
var te,i,j,cl:integer;
   p:array[1..50] of integer;
   arret:boolean;
begin
    for i:=1 to tl do
       m[i,tc]:=0;
    arret:=false;
    cl:=1;
    repeat
    begin
        j:=1;
        te:=0;
        while (m[j,tc]<>0)  and (j<=tl) do j:=j+1;
        if m[j,tc]<>0 then arret:=true
        else
        begin
            te:=te+1;
            p[te]:=j;
            for j:= p[te]+1 to tl do
               if m[j,tc]=0 then
                  if m[j,tc-1]>m[p[te],tc-1] then
                  begin
                      te:=1;
```

```
                    p[te]:=j;
                end
                else
                   if m[j,tc-1] = m[p[te],tc-1] then
                   begin
                       te:=te+1;
                       p[te]:=j;
                   end;
            for i:=1 to te do
               m[p[i],tc]:=cl;
            cl:=cl+te;
         end;
    end;
    until arret;
end;
```

Exercice 62

Le tri par bulles est un tri par échange. Le principe de base est de réordonner les couples non classés tant qu'ils en existent. La méthode de tri par bulles consiste à parcourir la liste en comparant deux éléments successifs en les permutant s'il y a lieu. Ecrire une procédure qui réalise ce tri.

```
Type tab=array[0..50] of integer ;
Procedure echanger(var t:tab;i,j: integer);
Var tmp:integer;
Begin
  tmp :=t[i];
  t[i] :=t[j];
  t[j] :=tmp;
End;
Procedure tri_bulles(var t: tab; n: integer);
Var p,k:integer;
Begin
  for k:=0 to n-2 do
   for p:=n-2 downto k do
     if t[p+1]<t[p] then
       echanger(t,p+1,p);
End;
```

Exercice 72

Ecrire une fonction qui retourne le déterminant d'une matrice carrée.

```
const DIM=10;
type matrice=array[0..DIM-1,0..DIM-1] of real;
```

```
procedure det_aux(ma:matrice;var mb:matrice;l,c:integer;n:integer);
var i,j,d,e:integer;
begin
     e:=0;
    for i:=0 to n-1 do
    begin
     d:=0;
        if i<>l then
        begin
        for j:=0 to n-1 do
          if j<>c then
               begin
               mb[e][d]:=ma[i][j];
               d:=d+1;
              end;
         e:=e+1;
        end;
  end;
end;

function expo(n:integer):real;
begin
 if n mod 2=0 then expo:=1
 else expo:=-1;
end;

function determinant(m:matrice;l:integer):real;
var
  i:integer;
  m2:matrice;
  x:real;
begin
 x:=0;
 if l=1 then determinant:= m[0][0]
 else
 begin
  for i:=0 to l-1 do
  begin
         det_aux(m,m2,i,0,l);
         x:=x+(expo(i)*m[i][0]*determinant(m2,(l-1)));
```

```
  end;
  determinant:=x;
 end;
end;
```

Exercice 73

Ecrire une procédure qui calcule l'inverse d'une matrice carrée.

```
procedure transp_mat(ma:matrice;var mb:matrice;n:integer);
var i,j:integer;
begin
for i:=0 to n-1 do
 for j:=0 to n-1 do
    mb[j][i]:=ma[i][j];
end;

procedure multi_R(a:real;ma:matrice;var mb:matrice;n:integer);
var i,j:integer;
begin
 for i:=0 to n-1 do
     for j:=0 to n-1 do
     mb[i][j]:=ma[i][j]*a;
end;

procedure coffacteur(ma:matrice;var mb:matrice;l:integer);
var i,j:integer;
   m2:matrice;
begin
 if l=1 then
         mb[0][0]:=1
 else
 begin
      for i:=0 to l-1 do
       for j:=0 to l-1 do
       begin
        det_aux(ma,m2,i,j,l);
        mb[i][j]:=expo(i+j)*determinant(m2,(l - 1));
       end;
 end;
end;

procedure inverse(ma:matrice;var mb:matrice;l:integer);
```

```
var
 m1,m2:matrice;
 d:real;
begin
 d:=(1/determinant(ma,l));
 coffacteur(ma,m1,l);
 transp_mat(m1,m2,l);
 multi_R(d,m2,mb,l);
end;
```

Exercice 74

Un carré magique est un carré divisé en cellules dans lesquelles les nombres entiers, à partir de 1 sont disposés de telle sorte que les sommes de chaque ligne, de chaque colonne et de chaque diagonale soient égales.

Exemple :

6	1	8
7	5	3
2	9	4

Ecrire une procédure qui permet de réaliser le carré magique.

```
const k=11;
type magique=array[1..k,1..k]of integer;
        procedure CaMag(var carre:magique;n:integer);
var fin,nombre,l,c:integer;
begin
    nombre:=1;
    l:=1;
    fin:=n*n;
    c:=(n+1) div 2;
    carre[l,c]:=nombre;
    while nombre<>fin do
    begin
        if nombre mod n =0 then inc(l)
          else
             begin
                 if l=1 then l:=n
                    else dec(l);
                 if c=1 then c:=n
                    else dec(c);
             end;
        inc(nombre);
        carre[l,c]:=nombre;
```

```
    end;
end;
```